Critical Infrastructure 
Software Security: 
A Maritime Shipping Study Case

Barton P. Miller
Computer Sciences Department
University of Wisconsin
bart@cs.wisc.edu

Elisa Heymann
Computer Sciences Department
University of Wisconsin
Universitat Autònoma de Barcelona
elisa@cs.wisc.edu

O’Reilly Velocity’18
Oct. 30-Nov. 2, 2018, London
Context

Container terminals have a strong dependency on software.
Problem

• Computer systems that control maritime shipping are at risk due to the software they use.

• The software has vulnerabilities, and is therefore open to cyber-attacks.

• Terminal Operating Systems (TOS) and Port Community Systems (PCS) are especially critical.

• The cost of a disruption is at least $1 billion/day and has a cascade effect.
Good work in risk assessment, but …

• It’s only a start.
• We need to focus on the software systems themselves (TOS, PCS).
• Only through an in-depth assessment of the software, can we be confident in its security.

We are addressing that challenge!
Our Work

• We started an effort to perform an in-depth vulnerability assessment of a TOS/PCS.

• First and critical step: have a software provider involved.
  – Social and psychological challenges to recognize the problem.
  – Surprisingly, we were given access to all their software technology.
How Did It Happen?

• Our first observations,
How Did It Happen?

• Our first observations,
How Did It Happen?

• Our first observations,
• ... to false steps,
• ... to meetings with FEPORTS, Valencia,
• ... to meetings with NOATUM, Valencia,
How Did It Happen?

• ... to meetings with NOATUM, Valencia,
How Did It Happen?

• Our first observations,
• ... to false steps,
• ... to meetings with FEPORTS, Valencia,
• ... to meetings with NOATUM, Valencia,
• ... to contacts with a software provider and establishing trust,
• ... to having access to the software and carrying out the actual assessment.
What Did We Do?

Looked for vulnerabilities in the TOS/PCS

What is a vulnerability?

“A vulnerability is a defect or weakness in system security procedures, design, implementation, or internal controls that can be exercised and result in a security breach or violation of security policy.”

- Gary McGraw, *Software Security*
What Did We Do?

We only cared about vulnerabilities we could exploit.

What is an exploit?

“The process of attacking a vulnerability in a program is called exploiting.”

The Art of Software Security Assessment
What Did We Do?

• Assessed a couple of software modules providing: Terminal Monitoring, Electronic Document Interchange (EDI) services, and movement of containers in the yard.

• Web-based system providing interface to current operation details of entire port, including gates, yards, ships, preadvice, containers, dangerous cargo, and related schedules and statuses.
How Did We Do it?

- First Principles Vulnerability Assessment (FPVA).
- While this takes time and effort, it’s the only way to achieve strong security.
- FPVA Focuses on critical assets.
- Is not based on known vulnerabilities.
How Did We Do it?

FPVA:

Step 1: Architectural Analysis
Step 2: Resource Identification
Step 3: Trust & Privilege Analysis
Step 4: Component Evaluation
Step 5: Dissemination of Results
How Did We Do it?

Client Browser

Application Server

Request

Response

DB
How Did We Do it?

Client Browser

Intercepting Proxy

Application Server

Request

Response

Request to attack server

Response

DB
What Did We Find?

There were problems in the software:

1. HTTP traffic was not encrypted.
   - Session hijacking.
   - Password sniffing.
   - Observing the network traffic to gain info of the port’s content without accessing the system.

2. Passwords were encrypted, not hashed.
Password/Traffic Sniffing

Unencrypted traffic visible to anyone on the network.

Client Browser

Login Request
username=administrator
password=pa$$w0rD

Response

Server

DB

Attacker
Session Hijacking

Unencrypted traffic visible to anyone on the network.
What Did We Find?

There were problems in the software:

3. Improper access to the database due to design issues, mostly validations only on the client side.

- As a consequence any user could change any other user’s password.
- Trust boundary problem.
- Design issues are expensive to fix.
Trust Boundary Violation

- Client is never to be trusted.
- Client is easy to replace or compromise.
- Any validation, authorization, or authentication on the client must be rechecked on the server.
Trust Boundary Violation

Client Requests Password Change for Currently Authenticated User

```java
... request.addParameter("username", currentUser.getUserName());
    request.addParameter("newPass", form.getNewPasswordField());
    httpClient.executeMethod(request);...
```

Attacker Modifies Request Data

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>realUser admin</td>
</tr>
<tr>
<td>newPass</td>
<td>password1</td>
</tr>
</tbody>
</table>

Server **Trusts** the Username and Handles the Request

```java
... username = request.getAttribute("username");
    newPass = request.getAttribute("newPass");
    userDB.updateRowPassword(username, newPass);
...```

---

**Trust Boundary Violation**

Client Requests Password Change for Currently Authenticated User

```java
... request.addParameter("username", currentUser.getUserName());
    request.addParameter("newPass", form.getNewPasswordField());
    httpClient.executeMethod(request);...
```

Attacker Modifies Request Data

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>realUser admin</td>
</tr>
<tr>
<td>newPass</td>
<td>password1</td>
</tr>
</tbody>
</table>

Server **Trusts** the Username and Handles the Request

```java
... username = request.getAttribute("username");
    newPass = request.getAttribute("newPass");
    userDB.updateRowPassword(username, newPass);
...```
What Did We Find?

There were problems in the software:

4. Use of vulnerable old version of some software frameworks.

- Software supply chain issues: libraries, underlying OS, compilers.
- Tools like OWASP Dependency Check, Dependabot, and Sonatype’s Application Health Check can help.
- Dynamic dependences and updates make this more difficult. Very hard issue.
What Did We Find?

There were problems in the software:

5. Users can modify and delete any files on the server machine.
   - Intercept a legitimate file request, then modify the request.
   - Improper validation allows path traversals.
The client specifies the name of a file for the server to delete. Without proper sanitation, the string '../' will traverse out of the specified directory. The server restricts file access to a specific directory by prepending that directory to the requested filename.

**Delete File Request**

file="..//Users/some_admin/important.doc"

**Successful Response**

C:\\safedir\\..//Users/some_admin/important.doc
Directory Traversal

C:/safedir/..\Users\some_admin\important.doc

C:
  Program Files/
  ProgramData/
  safedir/
    temp01.txt
    temp02.txt
    (...)
  Users/
    some_admin/
      important.doc
  Windows/
Directory Traversal

1. Request: file="../Users/some_admin/important.doc"

```java
String path = request.getParameter("file");
// check for dir separators to prevent escape from safedir
if(path.contains(java.io.File.separator)){
    throw new PathTraversalException(path + " is invalid.");
}
path = "C:\\safedir\\" + path;
File f = new File(path);
f.delete();
```

2. Server deletes C:\Users\some_admin\important.doc

Separators predefined:
- on Windows java.io.File.separator = "\\"
- on Unix java.io.File.separator = "/

Java File() constructor adapts pathname to underlying OS.
Then What?

- We suggested remediations to the software provider.
- We reviewed the code after the remediations.
- Several rounds of interactions were needed to implement the right fixes.
- They had an urgent need for training in software assurance and secure programming. Accomplished.
Closing Thoughts

• The TOS and PCS are large and complex pieces of software.
• No one has previously carried out an in-depth assessment of a TOS or PCS.
• An in-depth vulnerability assessment of the TOS and PCS is essential to prevent cyber-attacks.
• The vulnerabilities are there. Who will exploit them first?
• The involvement of software providers is essential.
Questions?